Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation
CRT-2006-16

FAST HEURISTICS FOR THE STEINER TREE PROBLEM WITH REVENUES, BUDGET AND HOP CONSTRAINTS

by

Alysson M. Costa
Jean-François Cordeau
Gilbert Laporte

July 2006

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone: (514) 343-7575 / Télécopieur: (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC ÉCOLE POLYTECHNIQUE MONTRÉAL Université de Montréal

PUBLICATION
CRT-2006-16

FAST HEURISTICS FOR THE STEINER TREE PROBLEM WITH REVENUES, BUDGET AND HOP CONSTRAINTS

by

Alysson M. Costa[1]
Jean-François Cordeau[1]
Gilbert Laporte[1]

This research was supported in part by the Brazilian Federal Agency for Post-Graduate Education – CAPES, and by the Natural Sciences and Engineering Research Council of Canada (NSERC).

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7, and Canada Research Chair in Distribution Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7 (alysson, cordeau, gilbert@crt.umontreal.ca)

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2006

106-06869/

Abstract

This article describes and compares three heuristics for a variant of the Steiner tree problem with revenues, which includes budget and hop constraints. First, a greedy method which obtains good approximations in short computational times is proposed. This initial solution is then improved by means of a destroy-and-repair method or a tabu search algorithm. Computational results compare the three methods in terms of accuracy and speed.

Keywords: network design, Steiner tree problem, budget, hop constraints, heuristics, tabu search.

Résumé

Cet article décrit et compare trois heuristiques pour une variante de problème de Steiner avec revenus, budget et contraintes d'éloignement. D'abord, une méthode gloutonne qui obtient de bonnes solutions en de courts temps de calcul est proposée. Cette solution est ensuite améliorée à l'aide d'une méthode de destruction-et-réparation ou avec un algorithme de recherche avec tabous. Des tests comparent les trois méthodes en termes d'efficacité et de vitesse.

Mots-clés: conception de réseaux, problème de Steiner, budget, contraintes d'éloignement, heuristiques, recherche avec tabous.

1 Introduction

Network design problems arise in a large variety of real life situations. In particular, spanning trees are used to model and solve planning problems frequently encountered in telecommunications (Voß, 2006) and electricity distribution (Carneiro et al., 1996; Avella et al., 2005).

The purpose of this article is to develop heuristics for the *Steiner Tree Problem with Revenues, Budget and Hop Constraints* (STPRBH). This problem is a generalization of several well-studied problems such as the *Minimum Spanning Tree Problem* (MSTP) and the *Steiner Tree Problem* (STP) described as follows. Let $G = (V, E)$ be a graph with vertex set $V = \{1, \ldots, n\}$, where vertex 1 is the *root* vertex, and edge set $E = \{(i, j) : i, j \in V, i < j\}$. In what follows, (i, j) must be interpreted as (j, i) if $i > j$. Each vertex has an associated revenue $r_i \geq 0$, and each edge has an associated cost $c_{ij} \geq 0$. Also, let T be a tree in G, and let $v(T)$ and $e(T)$ be the sets of vertices and edges belonging to T, respectively. The STPRBH is the problem of determining a tree T of maximal revenue $r(T) = \sum_{i \in v(T)} r_i$, rooted at vertex 1, and subject to two constraints: a budget constraint limiting the network cost $\sum_{(i,j) \in e(T)} c_{ij}$ to a maximum value b, and hop constraints limiting the number of edges h_i between $i \in v(T)$ and the root vertex to a maximum value h.

The STP is a well-studied problem and so is its version with revenues. The latter problem is useful in situations where one must decide whether or not to serve certain vertices. The idea is that one is not obliged to serve all vertices as in the MST, or specified vertices as in the STP. Instead, the number and location of the vertices to be served are part of the decision problem. For example, a cable television company planning to expand its network can decide whether or not to serve a certain area, based on the combination of the expected revenues that will come from the clients in that area and the network expansion cost. Steiner tree problems with revenues have been used to solve design problems in local access networks (Canuto et al., 2001; Cunha et al., 2003), and in heating networks (Ljubić et al., 2005). In network expansion problems, one is usually faced with budgetary constraints that come from the fact that a limited investment budget is available for a certain area or a certain period of time. Although it is quite easy to understand the practical aspect of these constraints, budget considerations are almost always ignored when dealing with STP with revenues (Costa et al., 2006b).

Concerning the hop constraints, their inclusion enables the consideration of reliability or transmission delays in the network. LeBlanc et al. (1999) have investigated the effect of hop constraints both on the reliability and on the maintenance of the signal quality in packet-switched telecommunication networks. Gouveia and Magnanti (2003) also use hop constraints to guarantee quality of service in telecommunications networks. Other uses of hop constraints have also been proposed in the literature. Balakrishnan and Altinkemer (1992) have imposed hop

constraints to generate alternative communications networks, while Voß (1999) has mentioned the utility of these constraints to model time durability constraints when solving some classes of lot-sizing problems by means of spanning tree models.

In this article we propose heuristic methods for the STPRBH. A previous study (Costa et al., 2006a) has shown the limits of exact algorithms for this problem. The heuristic methods proposed here are intended to obtain good approximations for large scale problems for which exact algorithms fail to converge. Several exact and heuristic methods already exist for the STP with profits (Costa et al., 2006b) and for spanning problems with hop constraints but, to our knowledge, no one has ever proposed approximate algorithms for the more general STPRBH. The closest problems solved by means of heuristic methods seem to be the MST and the STP with hop constraints. Lagrangian relaxation heuristics have been proposed for the MST with hop constraints (Gouveia, 1995; Gouveia, 1996; Gouveia, 1998; Gouveia and Requejo, 2001) and for the STP with hop constraints (Gouveia, 1998), while Voß (1999) has developed a greedy method and a tabu search algorithm for the hop constrained STP.

We first propose a fast greedy algorithm which is later followed by one of the following two improvement phases: a simple local search that destroys part of the solution and rebuilds it in a greedy fashion, or a more flexible tabu search algorithm that explores the solution space by means of two simple neighborhoods. In the remainder of this article we describe these three algorithms in detail. In the next section, we provide a mathematical formulation for the STPRBH in order to help understand the ideas behind the heuristics. Then, the algorithms themselves are described. The greedy and the destroy-and-repair approaches are presented in Sections 3 and 4, respectively. Section 5 describes the tabu search algorithm while computational results are provided in Section 6. The paper ends with some conclusions in Section 7.

2 Mathematical formulation

We first recall the undirected Dantzig-Fulkerson-Johnson model for the STPRBH. Constraints (4) and (5) of this model are useful to understand our heuristics. Let x_{ij} and y_i be binary variables associated with edges $(i,j) \in E$ and vertices $i \in V$, respectively. Variable y_i is equal to 1 if vertex i belongs to the solution ($y_1 = 1$), and to 0 otherwise. Similarly, variable x_{ij} is equal to 1 if edge (i,j) belongs to the solution, and to 0 otherwise. For $S \subseteq V$, define $E(S)$ as the set of edges with both end vertices in S. Let also $P = (i_1 = 1, \ldots, i_\ell)$ denote a path originating at the root vertex and containing ℓ vertices. Finally, define $\mathcal{P}_h$ as the set of paths P with $\ell \geq h + 2$, i.e., paths violating the hop constraint. The STPRBH can then be written as:

$$\text{Maximize} \sum_{i \in V} r_i y_i \tag{1}$$

subject to

$$\sum_{(i,j) \in E} x_{ij} = \sum_{i \in V} y_i - 1, \tag{2}$$

$$\sum_{(i,j) \in E(S)} x_{ij} \leq \sum_{i \in S \setminus \{k\}} y_i, \qquad k \in S \subseteq V,\ |S| \geq 2, \tag{3}$$

$$\sum_{(i,j) \in E} c_{ij} x_{ij} \leq b, \tag{4}$$

$$\sum_{t=2}^{\ell} x_{t-1,t} \leq h, \qquad P = (i_1 = 1, \ldots, i_\ell) \in \mathcal{P}_h, \tag{5}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in E, \tag{6}$$

$$y_1 = 1, \tag{7}$$

$$y_i \in \{0,1\}, \qquad i \in V \setminus \{1\}. \tag{8}$$

The goal is to maximize the collected revenues while respecting the budget and hop constraints, (4) and (5), respectively. Constraint (2) guarantees that in the solution the number of edges is equal to the number of vertices minus one, and constraints (3) are the connectivity constraints.

3 Greedy algorithm

The idea of our greedy algorithm is to iteratively construct a rooted tree while maintaining the feasibility of constraints (4) and (5). We start with a solution containing only the root vertex. At each step of the algorithm, one adds a path connecting a non-selected profitable vertex to the existing solution. The algorithm stops when it cannot find a profitable vertex that can be added to the solution without incurring a violation of either the budget or the hop constraints. Two basic steps are needed to implement this idea. First, one must be able to produce feasible paths connecting non-selected profitable vertices to the existing solution. Second, one must evaluate these paths and choose the best one (in a greedy sense). These two steps are now detailed.

3.1 Finding feasible paths

Adding at each step of the algorithm a path not creating cycles to the existing solution guarantees the satisfaction of constraints (2) and (3). In order to respect the budget constraint,

the sum of the edge costs in the added path plus the costs of the edges already in the solution must not exceed the budget value. Therefore, one has interest in finding shortest paths connecting profitable vertices to the current tree. On the other hand, the number of the edges in a path must be limited, in order to respect the hop constraints. It is important to note that it does not suffice to guarantee that the added paths have less than h edges. Instead, one must also ensure that the number of edges in the complete path from the added profitable vertex to the root vertex (which might include edges that were already in the existing solution) is limited to h.

Based on these two considerations, it is easy to conclude that a building block for constructing feasible paths is the hop constrained shortest path problem, i.e., the problem of determining a shortest path between two vertices containing at most h edges. This problem can be solved efficiently by dynamic programming (Lawler, 1976). Let $L(i,j,\ell)$ represent the cost of a shortest path between an origin vertex i and a destination vertex j, containing at most ℓ edges. Then

$$\begin{aligned}
&L(i,i,0) = 0, && i \in V,\\
&L(i,j,0) = \infty, && i,j \in V, j \neq i,\\
&L(i,j,\ell) = \min\{L(i,j,\ell-1), \min_{k|(k,j)\in E}\{L(i,k,\ell-1) + c_{kj}\}, && i,j \in V, \ell \geq 1.
\end{aligned}$$

Note that, contrary to what happens in the shortest path problem, an optimal subpath between two vertices is dependent on the complete path to which it belongs. This happens due to the hop constraints, as shown in Figure 1. Costs are shown next to the edges. The figure depicts optimal shortest paths from vertex 1 to the other vertices in the graph, with a hop limit of 3. Note that the optimal subpath from vertex 1 to vertex 4 includes vertices 2 and 3 if vertex 4 is the destination, but includes only vertex 6 if vertex 5 is the destination. In implementation terms, this means that one must keep a precedence array for each vertex of the graph vertices while solving the problem.

With an algorithm for the shortest path problem with hop constraints, one can evaluate the cost of connecting the non-selected profitable vertices to the existing tree and develop a simple algorithm. Let T be the current solution. At the first iteration, $v(T) = \{1\}$ and the candidate destination vertices j are those with $L(1,j,h) \leq b$ and $r_j > 0$. Note that if a path violates the hop constraint, the hop constrained shortest path will return ∞ and, therefore, this path will not be considered for insertion in the solution. When at least one path has been added, one must not only consider the new paths that originate at the root vertex, but also all paths that are originated at other vertices in $v(T)$. The algorithm considers for insertion all shortest paths such that

$$L(i,j,h-h_i) \leq b - \sum_{(i,j)\in e(T)} c_{ij}, \qquad i \in v(T), j \notin v(T), r_j > 0.$$

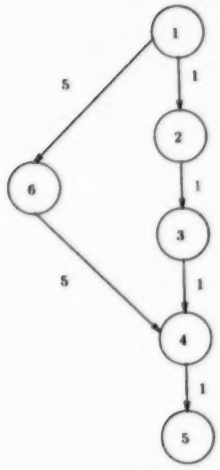

Figure 1: Example of different optimal subpaths between two vertices depending on the complete path.

Instead of running the hop constrained shortest path algorithm for all vertices in the current solution, one can simply set the costs of the edges in $e(T)$ to 0 and run the algorithm a single time, with the root vertex as the origin. Because of the dynamic programming structure, a single run is able to compute the hop constrained shortest distances from the root to all the other vertices, saving computation time.

3.2 Evaluating the paths

At each step, a single new path is added to the solution. The criterion used to choose the path to be added is an important part of the algorithm. If one makes a choice based only on profits, there is a risk of adding very costly paths that will soon consume the whole budget. On the other hand, if one chooses to add shortest paths, some vertices with a high revenue but at a moderate distance from the current solution may be ignored. Based on preliminary tests, we found that the best strategy was to consider both the costs and the revenues when deciding on the path to be added. We tested several ratios r_j^α / c^β, where α and β are two non-negative numbers, r_j is the revenue of the profitable vertex to be added, and c is the sum of the cost of the edges in the path. We found that the best parameter setting was $\alpha = 3$ and $\beta = 1$, which was used in all subsequent tests.

Algorithm 1, described in the following figure, presents the complete greedy algorithm in details. A few comments on this algorithm are in order. First, note that when adding a path one

Algorithm 1 Greedy algorithm for the STPRBH

T is initially the root node: $v(T) = \{1\}$.
loop
  $\overline{p} = 0, \overline{c} = 0, \overline{j} = -1$.
  for all vertices $j \notin v(T)$ **do**
    $c = L(1, j, h)$;
    $p = r_j^3/c$.
    if $\sum_{(i,j)\in e(T)} c_{ij} + c > b$ **then**
      $p = 0$.
    end if
    if $p \geq \overline{p}$ **then**
      $\overline{j} = j$;
      $\overline{c} = c$;
      $\overline{p} = p$.
    end if
  end for
  if $\overline{p} = 0$ **then**
    END
  end if
  P is the path from 1 to $\overline{j}$: $P = (i_1 = 1, \ldots, i_\ell = \overline{j})$;
  $c_{i_t, i_{t+1}} = 0$, $t = 1, \ldots, \ell - 1$;
  $v(T) = v(T) \cup v(P)$;
  destroy possible cycles.
end loop
Return T.

can also add other profitable vertices in addition to the end vertex. It has proved to be slightly better not to consider these profits when deciding which path should be added. This way, the algorithm tends to add shorter paths first, thus reducing the risk of making wrong decisions due to the myopic character of the choices. Second, because of the hop constraints, one can sometimes introduce cycles when adding a path. Consider again Figure 1 and the situation where $h = 3$, all vertices are profitable and, after the first iteration of the greedy algorithm $v(T) = \{1, 2, 3, 4\}$. Assuming that the budget constraint is not violated, the algorithm will try to add vertex 5 to the solution, using the edges $(1, 6)$ and $(5, 6)$ in order to respect the hop constraint and therefore forming a cycle. When this situation happens, the algorithm destroys the cycle by inspecting vertices with two incoming edges (in this case, vertex 4) and eliminating

the edge that was inserted first (in this case, edge $(3, 4)$). This way, the cycle is eliminated and the hop constraints are respected for the entire graph.

4 Destroy-and-repair algorithm

The idea of destroying part of the solution and reconstructing it in a different way in order to obtain different and hopefully better solutions has already been used to solve other combinatorial problems. Ropke and Pisinger (2006) have employed such a technique to solve vehicle routing problems with time windows, and Pisinger and Ropke (2007) have applied it to a more general class of routing problems. Voß (1999) has successfully used a similar idea to solve the STP with hop constraints. For the STPRBH, we propose a very simple procedure that consists of sequentially blocking part of the current solution and rerunning the greedy algorithm.

A feasible solution to the STPRBH consists of a tree. Given an initial solution, we simply set the cost of some edges to infinity, one at a time, and we rerun the greedy algorithm with the modified costs. This is done for all edges of the current solution that are incident to a leaf vertex, as shown in Figure 2, and the best obtained solution is retained. The procedure is repeated for the new solution until there is no gain obtained with the blocking of a single edge. **Algorithm 2** schematically presents the procedure.

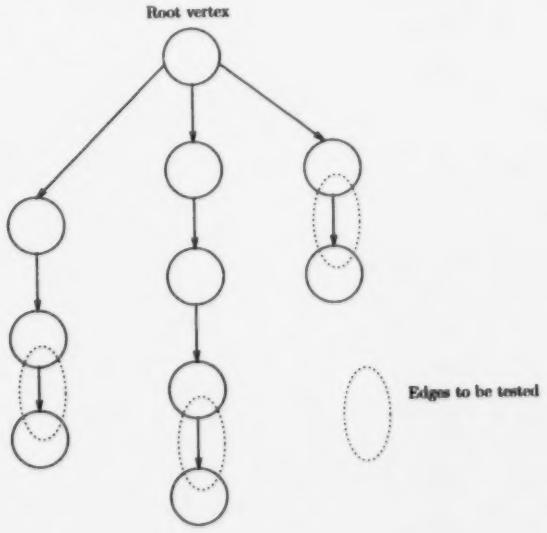

Figure 2: Edges incident to leaf vertices.

Other ways of preventing the greedy algorithm from repeatedly reaching the same solution

Algorithm 2 Destroy-and-repair algorithm for the STPRBH

T = solution of **Algorithm 1**;
best solution $\overline{T} = T$.
loop
  $\overline{r} = r(\overline{T})$.
  for all the edges (i,j) incident to a leaf vertex do
    store original cost: $\bar{c}_{ij} = c_{ij}$;
    $c_{ij} = \infty$;
    T = solution of **Algorithm 1** (with modified costs).
    if $r(T) > r(\overline{T})$ **then**
      $\overline{T} = T$.
    end if
    Restore original cost: $c_{ij} = \bar{c}_{ij}$.
  end for
  if $r(\overline{T}) = \overline{r}$ **then**
    END
  end if
end loop
$T = \overline{T}$;
return T

can be used. For example, one could block all edges in the solution, one at a time. We tested this idea and, although it was much more time consuming, the results were not much better than the ones obtained with **Algorithm 2**. In a way this was expected, since the greedy algorithm has a tendency to reconstruct a similar solution if an intermediate edge has been blocked. By concentrating on the edges incident to profitable leaf vertices we increase the probability of really blocking that solution and obtaining something new.

We also tested other neighborhoods such as blocking a vertex of the solution (by setting the cost of all its incident edges to infinity). The results turned out to be worse than those obtained with the original algorithm. In this case, we think that the explanation comes from the fact that blocking a vertex is probably too restrictive and by doing so we may eliminate several good solutions.

5 Tabu search algorithm

A tabu search algorithm was also developed for the STPRBH. The main idea of tabu search is to define a neighborhood for a solution and let the algorithm explore the search space by moving to the best allowed neighbor solutions, while defining some forbidden or tabu moves in order to avoid cycling.

We define too basic moves for the STPRBH. The first is an ***add*** move. It consists of adding a path originating at a vertex of the current solution and ending at a profitable vertex. This is very similar to the basic step of the greedy algorithm presented in Section 3, since this move also makes use of the shortest path with hop constraints procedure and, therefore, maintains hop-feasibility during the whole search. In order to increase the flexibility of the algorithm, we let the ***add*** move violate the budget constraint. The idea is to design an algorithm capable of exploring infeasible parts of the search space in the hope of reaching feasible solutions that could not be reached otherwise. To encourage the identification of feasible solutions during the search, a penalty is associated with trees that violate the budget. The idea of letting the tabu search explore infeasible areas of the search space has been used successfully in other combinatorial problems (see, e.g., Gendreau et al., 1994).

The second move is ***remove***, which consists of the deletion of a branch of the solution. It is the counterpart of the ***add*** move in the sense that the ***add*** move is driven by revenue collection, while the ***remove*** move is driven by budget feasibility. In the following, the two moves are described in detail. A third move, which is called ***destroy*** is also described. We have used this move only occasionally to help the search algorithm escape from local optima.

5.1 Add move

An ***add*** move is like a single step of the greedy algorithm described in Section 3: one adds the hop constrained shortest path connecting the best non-selected vertex to the current tree. This is done by setting the edge costs in the current solution to 0 and running the hop constrained shortest path procedure, which returns the hop constrained shortest paths originating at the root and ending at each vertex of the graph. Once a vertex has been added, it is declared tabu for a number of iterations. This way, the branch connecting the previous solution to the newly added vertex will not be deleted by the ***remove*** move until its tabu status has been lifted.

The tabu mechanism helps avoid cycling and forces the algorithm to explore new areas of the search space, differentiating the tabu search from the greedy algorithm presented earlier. Another strategy is used to increase the diversification of the solutions. At each iteration, one considers only a subset of randomly selected vertices: the algorithm arbitrarily chooses a subset of the possible vertices and adds the best one. Note that this is not done to save any

computational time since the needed information was already available from the hop constrained shortest path procedure even for the discarded vertices, but as a diversification mechanism.

5.2 Remove move

The ***remove*** move consists of eliminating branches of the current solution. It was made necessary as a counterpart of the ***add*** move in order to help the algorithm identify budget-feasible solutions. We carefully select the branches to be eliminated in order to introduce only local perturbations in the current solution. This is done by connecting a vertex with degree larger or equal to three to a leaf of the current tree. We call the vertices with degree larger or equal to three *breaking points*, since it is at these vertices that the solution is cut. Figure 3 depicts an example of a tree solution. The breaking points are highlighted while the possible paths to be cut are shown in dotted lines.

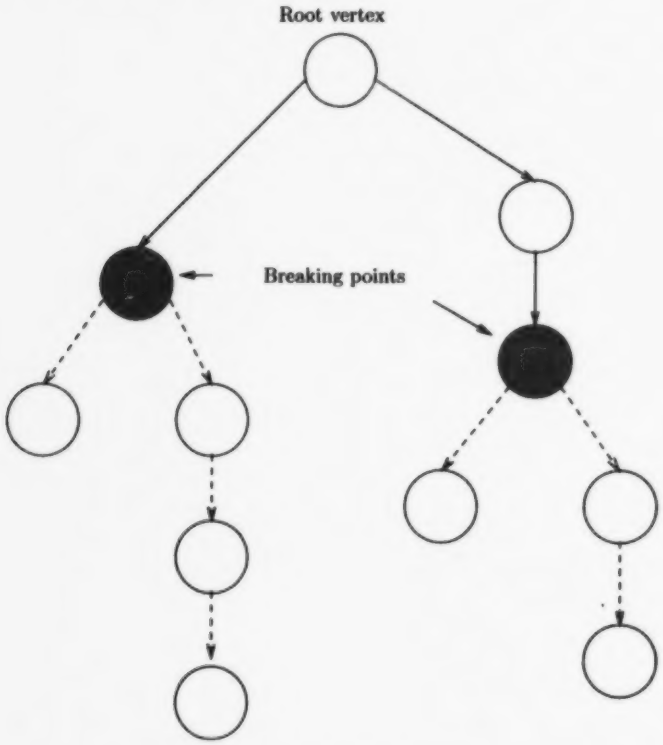

Figure 3: Breaking points and possible paths to be removed

5.3 Destroy move

A ***destroy*** move that erases a whole subtree of the current solution has also been developed. This move acts by randomly selecting a vertex in the current solution and deleting the whole subtree rooted at that vertex. This move was only used after the algorithm spent a certain number of iterations without improving the best known solution value.

The three moves just described have been incorporated within the tabu search algorithm. The ***add*** and ***remove*** moves are evaluated together. First, the gain of all ***add*** moves is computed. The gain of one ***add*** move connecting the profitable vertex i to the solution is given by the value of the revenue of the added vertex, minus the difference between the penalties after and before the move:

$$\gamma^{+} = r_i - \pi(T') + \pi(T''),$$

where $\pi(T)$ is the penalty associated with a solution T, and T' and T'' are the solutions before and after the move, respectively. For a given solution T with total network cost $\sum_{(i,j)\in e(T)} c_{ij}$, the penalty is given by:

$$\pi(T) = \begin{cases} 0, & \text{if} \quad \sum_{(i,j)\in e(T)} c_{ij} \leq b \\ \phi \times \left[\sum_{(i,j)\in e(T)} c_{ij} - b\right], & \text{if} \quad \sum_{(i,j)\in e(T)} c_{ij} > b, \end{cases}$$

where ϕ is a penalty parameter updated dynamically during the search.

In a complementary manner, the ***remove*** move yields a gain equal to:

$$\gamma^{-} = -r_i - \pi(T') + \pi(T'').$$

At each iteration, the move with the best gain is performed and the penalty parameter ϕ is updated. If the new solution is budget feasible, ϕ is halved, thus encouraging the algorithm to look for budget-infeasible solutions. Similarly, if the new solution is budget-infeasible, ϕ is doubled.

Algorithm 3 presents the tabu search algorithm in detail. It stops after a preset number of iterations.

6 Computational experiments

We have tested the algorithms just described on several instances obtained from the OR-Library (Beasley, 1990). The STP instances contain two set of vertices: *mandatory* vertices which must be present in the solution, and *Steiner* vertices which can be used if their inclusion in the solution reduces the total cost. We have adapted these instances for the STTPBH by using the mandatory vertices as profitable vertices with revenue given by a discrete uniform

Algorithm 3 Tabu search algorithm for the STPRBH

T = solution of **Algorithm 1**;
$\overline{T} = T$;
$f = 1$.
loop
  Randomly select a subset of the profitable vertices not in the tree and compute gain;
  compute gain for remove moves;
  perform move with best gain and update T;
  make move tabu for 5 iterations.
  if solution is budget-feasible **then**
    $\phi = \phi/2$.
    if $r(T) > r(\overline{T})$ **then**
      $\overline{T} = T$.
    end if
  else
    $\phi = 2\phi$.
  end if
  if no improvement in more than 100 iterations **then**
    apply destroy move.
  end if
  if stopping criterion reached **then**
    END
  end if
end loop

distribution on 1 and 100. The Steiner vertices were attributed a zero revenue. Different values for the budget and for the hop limit were attributed to each instance, thus creating several scenarios.

Tables 3 to 5 show the complete results for the mid-sized instances for which the exact methods presented by Costa et al. (2006a) were able to converge within a time limit of two hours. In the tables, we present the size of the instances, the parameters of each scenario and the optimal solution value. We also provide the gap and the computation time for each of the proposed methods: the greedy algorithm, the destroy-and-repair algorithm, and the tabu search algorithm. For this algorithm, we have run two tests. In the first one, the tabu search was run for 2000 iterations, while in the second it was allowed to run five times longer, for 10,000 iterations.

We use these instances in order to evaluate the gap between the solutions obtained by the heuristics and the optimal solutions. Table 1 summarizes the results obtained for the first group of instances. In this table, for each method, we report the mean and maximum gap, as well as the mean and maximum computation times.

Method	Gap	Max Gap	Seconds	Max seconds
Greedy	2.50 %	31.97 %	< 0.01	0.02
Destroy-and-repair	1.13 %	10.10 %	0.09	1.44
Tabu (2000)	0.52 ± 0.08 %	12.99 ± 7.55 %	0.38	0.90
Tabu (10,000)	0.24 ± 0.05 %	8.46 ± 8.01 %	1.80	4.21

Table 1: Summary of results for the mid-sized instances

We first observe that even the simplest of the three algorithms can solve a large number of instances to optimality. Indeed, the greedy algorithm finds an optimal solution for 78 out of the 144 instances. Note that we did not consider instances with a budget equal to the total network cost since these instances are easy and the greedy algorithm always finds an optimal solution. It is clear, however, that for some instances the greedy algorithm can be trapped in a poor quality local optimum. In the worst of the cases, the algorithm generated a solution with an optimality gap almost equal to 32%. Finally, concerning the computational time, since the algorithm is based on quick shortest paths calculations, a maximum of 0.02 seconds was required to solve any instance.

Starting from the solution of the greedy algorithm, both the destroy-and-repair were able to reduce the optimality gaps. The first method reduced the mean gap from 2.50% to 1.13% and the maximum gap from 32% to 10.10%. The tabu search was able to further reduce these gaps. The mean gap (averaged over ten runs) was reduced to 0.52% when 2000 iterations were allowed, and to 0.24% when 10,000 iterations were allowed. The maximum gaps for these two cases were 12.99% and 8.46%.

To better evaluate the efficiency of the improvement routines, we consider a second group of larger instances with up to 500 vertices and 12,500 edges. For these instances, the exact algorithms described in Costa et al. (2006a) generally did not obtain solutions within the allowed time limit of two hours, often failing to even solve the linear relaxation at the root node. For this reason, we compare the results obtained by the destroy-and-repair and the tabu search methods to the results obtained by the greedy algorithm. The complete set of results is presented in Tables 6 to 10. In these tables we present the value of the solution found by the greedy algorithm and the percentage improvement obtained by the other methods. We also show the computational times for each instance. For the tabu search heuristic, the results are averaged

over ten runs (when 2000 iterations were executed) and three runs (when 10,000 iterations were executed).

Table 2 presents a summary of the results presented in Tables 6 to 10. We can see that even for large instances, the greedy algorithm is very fast, taking on average 2.32 seconds, with a maximum of 39.19 seconds. Still concerning the times, it is interesting to see how the destroy-and-repair can easily become very computationally inefficient, taking more than two hours to solve some of the instances. The tabu search, even when run for 10,000 iterations, is on average twice faster than the destroy-and-repair algorithm.

Method	Improvement	Max improvement	Seconds	Max seconds
Greedy	-	-	2.32	39.19
Destroy-and-repair	1.31 %	14.36 %	408.62	9074.30
Tabu (2000)	1.33 ± 0.06 %	17.86 ± 2.18 %	58.96	513.04
Tabu (10,000)	1.94 ± 0.02 %	23.92 ± 3.41 %	214.29	1513.30

Table 2: Summary of results for the larger instances

Concerning the improvements, the destroy-and-repair algorithm yields mean and maximum improvements of 1.31% and 14.36%, respectively. The tabu search can increase these percentages to 1.33% and 17.86% when run for 2000 iterations, and obtains considerably better solutions when 10,000 iterations are allowed. It then yields 1.94% average improvement in comparison to the greedy algorithm, and a maximum improvement of almost 24%.

7 Conclusions

We have proposed three heuristics for a modified version of the Steiner Tree Problem with revenues including budget and hop constraints. A former study (Costa et al., 2006a) has shown that exact algorithms based on existing formulations cannot be applied to large instances of this problem. The heuristics we have proposed are based on simple construction and tabu search mechanisms. The greedy algorithm is extremely fast but may be trapped in a local optimum due to its myopic nature. The destroy-and-repair method is a simple modification of the greedy algorithm which considerably improves its solutions, albeit at the cost of a large increase in computation time. Finally, the tabu search is capable of consistently identifying optimal solutions while maintaining the computational times at a reasonable level.

Acknowledgments

This work was supported by the Brazilian Federal Agency for Post-Graduate Education - CAPES, under grant BEX 1097/01-6 and by the Canadian Natural Science and Engineering Research Council under grants 227837-04 and OGP0039682. This support is gratefully acknowledged.

Instance						Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Optimum	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
MSteinb1	50	63	5	3	140	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.1	0.00 ± 0.00	0.44
			10	3	140	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.43
MSteinb2			5	3	182	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.43
			10	3	182	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.08	0.00 ± 0.00	0.38
MSteinb3			5	3	253	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.37
			10	3	253	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.07	0.00 ± 0.00	0.36
MSteinb4		100	5	3	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.06	0.00 ± 0.00	0.31
			10	3	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.06	0.00 ± 0.00	0.3
MSteinb5			5	3	588	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.07	0.00 ± 0.50	0.31
			10	3	565	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.68
MSteinb6			5	3	1035	6.28	0.00	2.90	0.02	0.53 ± 1.17	0.14	0.00 ± 0.86	0.68
			10	3	610	5.08	0.00	1.80	0.00	4.00 ± 1.45	0.14	0.43 ± 1.79	0.68
MSteinb7	75	94	5	3	87	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.16	0.00 ± 0.00	0.78
			10	3	87	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.78
MSteinb8			5	3	85	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.16	0.00 ± 0.00	0.78
			10	3	85	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.74
MSteinb9			5	3	596	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.16	0.00 ± 0.00	0.74
			10	3	483	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.18	0.00 ± 0.00	0.76
MSteinb10		150	5	3	319	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.44
			10	3	319	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.09	0.00 ± 0.00	0.44
MSteinb11			5	3	316	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.1	0.00 ± 0.00	0.42
			10	3	306	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.1	0.00 ± 0.00	0.44
MSteinb13			5	3	1169	0.00	0.00	0.00	0.03	0.00 ± 0.00	0.09	0.00 ± 1.50	0.43
			10	3	1017	2.36	0.00	1.28	0.03	0.21 ± 0.55	0.09	0.00 ± 1.39	0.44
MSteinb15	100	125	5	3	147	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.14	0.00 ± 0.00	0.64
			10	3	147	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.65
MSteinb14			5	3	263	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.14	0.00 ± 0.00	0.66
			10	3	263	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.2	0.00 ± 0.00	0.95
MSteinb15			5	3	1061	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.2	0.00 ± 1.00	0.95
			10	3	830	2.17	0.00	2.17	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.95
MSteinb16		200	5	3	479	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.87
			10	3	479	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.18	0.00 ± 0.50	0.87
MSteinb17			5	3	254	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.2	0.00 ± 0.00	0.87
			10	3	254	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.22	0.00 ± 0.00	0.93
MSteinb18			5	3	1298	0.00	0.00	0.00	0.06	0.00 ± 0.00	0.21	0.00 ± 3.00	0.93
			10	3	1132	5.04	0.00	4.33	0.06	1.42 ± 0.98	0.22	0.20 ± 2.61	0.95
MSteinb1	50	63	5	6	403	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.74
			10	6	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.15	0.00 ± 0.00	0.73
MSteinb2			5	6	491	2.65	0.00	0.00	0.01	2.65 ± 0.00	0.17	2.65 ± 1.31	0.72
			10	6	300	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.13	0.00 ± 0.00	0.59
MSteinb3			5	6	807	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.13	0.00 ± 0.50	0.59
			10	6	624	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.12	0.00 ± 0.00	0.6
MSteinb4		100	5	6	455	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.87
			10	6	447	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.2	0.00 ± 0.50	0.87
MSteinb5			5	6	666	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.19	0.00 ± 1.00	0.89
			10	6	643	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.26	0.00 ± 0.50	1.2
MSteinb6			5	6	1257	2.78	0.00	1.59	0.08	0.60 ± 0.44	0.25	0.40 ± 3.76	1.19
			10	6	839	2.50	0.00	2.50	0.01	1.47 ± 0.63	0.26	0.86 ± 0.35	1.18
MSteinb7	75	94	5	6	627	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.15	0.00 ± 0.50	0.76
			10	6	432	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.17	0.00 ± 0.00	0.77
MSteinb8			5	6	535	5.79	0.00	0.00	0.02	0.64 ± 1.34	0.17	0.00 ± 0.87	0.76
			10	6	346	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.26	0.00 ± 0.00	1.22
MSteinb9			5	6	1343	8.71	0.00	4.69	0.08	0.10 ± 0.31	0.26	0.00 ± 2.94	1.22
			10	6	816	7.97	0.00	6.50	0.02	0.00 ± 0.00	0.27	0.00 ± 1.00	1.23

Table 3: Results for instances MSteinb

Instance						Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Optimum	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
MSteinb10		150	5	6	702	0.00	0.00	0.00	0.04	0.00 ± 0.00	0.22	0.00 ± 2.00	1.07
			10	6	668	1.95	0.00	0.75	0.05	0.13 ± 0.22	0.23	0.00 ± 2.44	1.08
MSteinb11			5	6	[illegible]	0.00	0.00	0.00	0.07	0.00 ± 0.00	0.24	0.00 ± 3.50	1.09
			10	6	[illegible]	31.97	0.00	1.57	0.04	10.82 ± 10.10	0.21	6.16 ± 3.24	1
MSteinb12			5	6	1847	0.27	0.00	0.05	0.31	0.23 ± 0.09	0.21	0.05 ± 15.44	1.01
			10	6	1384	3.32	0.00	3.32	0.05	2.58 ± 0.86	0.21	0.87 ± 1.96	1.01
MSteinb13	100	125	5	6	674	0.15	0.00	0.00	0.07	0.00 ± 0.00	0.21	0.00 ± 3.50	0.93
			10	6	452	16.81	0.00	0.00	0.02	2.65 ± 0.00	0.2	1.59 ± 1.13	0.95
MSteinb14			5	6	977	0.72	0.00	0.00	0.06	0.72 ± 0.00	0.2	0.72 ± 2.78	0.96
			10	6	595	0.17	0.00	0.00	0.02	0.00 ± 0.00	0.28	0.00 ± 1.00	1.2
MSteinb15			5	6	1858	16.42	0.00	8.18	0.15	2.22 ± 1.94	0.26	0.12 ± 6.85	1.2
			10	6	1040	6.92	0.00	6.92	0.02	1.66 ± 1.78	0.28	0.50 ± 0.67	1.21
MSteinb16		200	5	6	840	0.00	0.00	0.00	0.09	0.00 ± 0.00	0.24	0.00 ± 4.50	1.2
			10	6	767	4.69	0.00	0.00	0.10	0.00 ± 0.00	0.25	0.00 ± 5.00	1.2
MSteinb17			5	6	1299	0.00	0.00	0.00	0.13	0.00 ± 0.00	0.26	0.00 ± 6.50	1.21
			10	6	1091	8.62	0.00	6.51	0.14	3.89 ± 0.97	0.27	2.87 ± 5.84	1.26
MSteinb18			5	6	2575	1.28	0.01	0.27	0.75	0.36 ± 0.20	0.28	0.17 ± 37.38	1.27
			10	6	1917	2.97	0.00	0.00	0.28	2.97 ± 0.00	0.27	1.78 ± 13.26	1.28
MSteinb1	50	63	5	9	431	0.46	0.00	0.46	0.01	0.00 ± 0.00	0.28	0.00 ± 0.50	1.3
			10	9	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.27	0.00 ± 0.00	1.29
MSteinb2			5	9	600	2.17	0.00	0.00	0.03	0.00 ± 0.00	0.28	0.00 ± 1.50	1.31
			10	9	300	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.25	0.00 ± 0.00	1.15
MSteinb3			5	9	924	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.25	0.00 ± 1.00	1.16
			10	9	649	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.25	0.00 ± 0.50	1.17
MSteinb4		100	5	9	455	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.33	0.00 ± 0.50	1.57
			10	9	455	1.76	0.00	1.54	0.02	1.54 ± 0.00	0.34	1.54 ± 0.87	1.64
MSteinb5			5	9	666	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.37	0.00 ± 1.00	1.64
			10	9	652	1.38	0.00	0.00	0.04	0.00 ± 0.00	0.38	0.00 ± 2.00	1.66
MSteinb6			5	9	1262	0.40	0.00	0.08	0.13	0.40 ± 0.00	0.39	0.30 ± 6.39	1.88
			10	9	903	0.89	0.00	0.89	0.02	0.89 ± 0.00	0.42	0.80 ± 0.82	1.66
MSteinb7	75	94	5	9	662	0.00	0.00	0.00	0.03	0.00 ± 0.00	0.41	0.00 ± 1.50	1.86
			10	9	432	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.41	0.00 ± 0.50	1.91
MSteinb8			5	9	781	0.79	0.00	0.53	0.10	0.37 ± 0.25	0.45	0.16 ± 4.87	1.93
			10	9	537	25.51	0.00	7.26	0.02	0.00 ± 0.00	0.38	0.00 ± 1.00	1.85
MSteinb9			5	9	1388	6.84	0.00	2.31	0.10	0.00 ± 0.00	0.39	0.00 ± 5.00	1.83
			10	9	816	3.43	0.00	0.37	0.03	0.00 ± 0.00	0.38	0.00 ± 1.50	1.82
MSteinb10		150	5	9	702	0.00	0.00	0.00	0.05	0.00 ± 0.00	0.44	0.00 ± 2.50	2.09
			10	9	694	1.87	0.00	1.87	0.04	0.45 ± 0.78	0.45	0.19 ± 1.78	2.17
MSteinb11			5	9	893	0.00	0.00	0.00	0.08	0.00 ± 0.00	0.46	0.00 ± 4.00	2.17
			10	9	855	3.04	0.00	2.34	0.09	2.11 ± 0.74	0.44	1.09 ± 3.89	1.88
MSteinb12			5	9	1806	0.91	0.00	0.84	0.39	0.80 ± 0.12	0.41	0.48 ± 19.27	1.91
			10	9	1401	7.14	0.00	2.64	0.12	1.38 ± 1.16	0.45	0.34 ± 5.54	1.92
MSteinb13	100	125	5	9	745	3.09	0.00	0.00	0.16	0.00 ± 0.00	0.48	0.00 ± 8.00	2.09
			10	9	465	9.89	0.00	6.24	0.03	0.00 ± 0.00	0.45	0.00 ± 1.50	2.15
MSteinb14			5	9	1033	5.32	0.00	0.39	0.09	3.00 ± 1.23	0.47	1.56 ± 3.62	2.2
			10	9	595	0.17	0.00	0.00	0.02	0.00 ± 0.00	0.52	0.00 ± 1.00	2.53
MSteinb15			5	9	1891	14.38	0.00	10.10	0.22	1.16 ± 2.05	0.53	0.00 ± 10.50	2.51
			10	9	1086	6.26	0.00	6.17	0.06	2.37 ± 1.93	0.52	0.00 ± 2.51	2.4
MSteinb16		200	5	9	840	0.00	0.00	0.00	0.14	0.00 ± 0.00	0.53	0.00 ± 7.00	2.46
			10	9	800	0.00	0.00	0.00	0.11	0.00 ± 0.00	0.54	0.00 ± 5.50	2.47
MSteinb17			5	9	1299	0.00	0.00	0.00	0.19	0.00 ± 0.00	0.51	0.00 ± 9.50	2.5
			10	9	1178	5.18	0.00	1.95	0.20	1.64 ± 1.04	0.52	0.42 ± 9.50	2.51
MSteinb18			5	9	2585	0.39	0.01	0.19	1.17	0.14 ± 0.09	0.59	0.04 ± 58.46	2.5
			10	9	1997	9.16	0.00	6.36	0.35	1.37 ± 0.78	0.51	0.33 ± 17.09	2.54
MSteinb1	50	63	5	12	431	0.46	0.00	0.46	0.01	0.00 ± 0.00	0.26	0.00 ± 0.50	1.29
			10	12	341	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.28	0.00 ± 0.00	1.31

Table 4: Results for instances MSteinb (continued)

Instance						Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Optimum	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
MSteinb2			5	12	600	2.17	0.00	0.00	0.03	0.00 ± 0.00	0.3	0.00 ± 1.50	1.3
			10	12	300	0.00	0.00	0.00	0.00	0.00 ± 0.00	0.29	0.00 ± 0.00	1.34
MSteinb3			5	12	931	2.58	0.00	0.75	0.06	0.75 ± 0.00	0.28	0.60 ± 2.79	1.37
			10	12	649	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.23	0.00 ± 0.50	1.28
MSteinb4		100	5	12	455	0.00	0.00	0.00	0.02	0.00 ± 0.00	0.27	0.00 ± 1.00	1.25
			10	12	455	1.76	0.00	1.54	0.03	1.54 ± 0.00	0.28	1.54 ± 1.22	1.33
MSteinb5			5	12	666	0.00	0.00	0.00	0.03	0.00 ± 0.00	0.29	0.00 ± 1.50	1.36
			10	12	652	1.38	0.00	0.00	0.04	0.00 ± 0.00	0.32	0.00 ± 2.00	1.51
MSteinb6			5	12	1262	0.00	0.00	0.00	0.08	0.00 ± 0.00	0.32	0.00 ± 4.00	1.51
			10	12	903	0.89	0.00	0.89	0.02	0.89 ± 0.00	0.32	0.89 ± 0.82	1.51
MSteinb7	75	94	5	12	662	0.00	0.00	0.00	0.04	0.00 ± 0.00	0.32	0.00 ± 2.00	1.49
			10	12	432	0.00	0.00	0.00	0.01	0.00 ± 0.00	0.32	0.00 ± 0.50	1.48
MSteinb8			5	12	832	0.72	0.00	0.72	0.06	0.72 ± 0.00	0.33	0.65 ± 2.79	1.48
			10	12	537	25.51	0.00	7.26	0.03	0.00 ± 0.00	0.35	0.00 ± 1.50	1.59
MSteinb9			5	12	1388	6.84	0.00	2.31	0.12	0.00 ± 0.00	0.34	0.00 ± 6.00	1.59
			10	12	816	3.43	0.00	0.37	0.04	0.00 ± 0.00	0.35	0.00 ± 2.00	1.58
MSteinb10		150	5	12	702	0.00	0.00	0.00	0.06	0.00 ± 0.00	0.45	0.00 ± 3.00	1.94
			10	12	694	1.87	0.00	1.87	0.05	0.40 ± 0.62	0.41	0.00 ± 2.34	1.95
MSteinb11			5	12	893	0.00	0.00	0.00	0.09	0.00 ± 0.00	0.41	0.00 ± 4.50	1.97
			10	12	855	3.04	0.00	2.34	0.11	2.34 ± 0.00	0.43	0.47 ± 5.13	2.07
MSteinb12			5	12	1866	0.27	0.01	0.21	0.53	0.17 ± 0.12	0.44	0.00 ± 26.45	2.09
			10	12	1401	3.28	0.00	2.64	0.16	1.16 ± 0.30	0.45	0.93 ± 7.61	2.11
MSteinb13	100	125	5	12	745	4.43	0.00	0.00	0.18	0.00 ± 0.00	0.45	0.00 ± 9.00	2.03
			10	12	465	9.89	0.00	6.24	0.04	0.00 ± 0.00	0.43	0.00 ± 2.00	2.05
MSteinb14			5	12	1038	1.54	0.00	0.00	0.13	1.01 ± 0.56	0.42	0.59 ± 6.14	2.06
			10	12	595	0.17	0.00	0.00	0.03	0.00 ± 0.00	0.48	0.00 ± 1.50	2.34
MSteinb15			5	12	1891	6.82	0.00	6.82	0.15	1.81 ± 1.63	0.47	0.01 ± 6.97	2.38
			10	12	1109	8.21	0.00	8.03	0.07	2.52 ± 2.16	0.5	1.38 ± 2.54	2.37
MSteinb16		200	5	12	840	0.00	0.00	0.00	0.17	0.00 ± 0.00	0.52	0.00 ± 8.50	2.37
			10	12	800	0.00	0.00	0.00	0.13	0.00 ± 0.00	0.49	0.00 ± 6.50	2.4
MSteinb17			5	12	1299	0.00	0.01	0.00	0.26	0.00 ± 0.00	0.51	0.00 ± 13.00	2.41
			10	12	1225	7.43	0.00	3.59	0.20	2.67 ± 1.76	0.48	1.22 ± 13.54	2.32
MSteinb18			5	12	2585	0.39	0.02	0.19	1.44	0.14 ± 0.09	0.51	0.02 ± 71.96	2.33
			10	12	1997	9.16	0.01	8.81	0.43	1.10 ± 0.92	0.51	0.27 ± 21.12	2.31

Table 5: Results for instances MSteinb (continued)

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc1_10	500	625	10	5	8	0.00	0.00	0.00	0.00 ± 0.00	2.62	0.00 ± 0.00	5.05
steinc1_100	500	625	10	5	71	0.00	0.00	0.00	0.00 ± 0.00	2.71	0.00 ± 0.00	5.03
steinc1_10	500	625	30	5	8	0.00	0.00	0.00	0.00 ± 0.00	2.61	0.00 ± 0.00	5.01
steinc1_100	500	625	30	5	71	0.00	0.00	0.00	0.00 ± 0.00	2.67	0.00 ± 0.00	5.01
steinc2_10	500	625	10	5	32	0.00	0.00	0.02	0.00 ± 0.00	7.78	0.00 ± 0.00	5.47
steinc2_100	500	625	10	5	328	0.00	0.00	0.02	0.00 ± 0.00	6.95	0.00 ± 0.00	5.33
steinc2_10	500	625	30	5	32	0.00	0.00	0.02	0.00 ± 0.00	7.79	0.00 ± 0.00	5.35
steinc2_100	500	625	30	5	328	0.00	0.00	0.02	0.00 ± 0.00	6.90	0.00 ± 0.00	5.46
steinc3_10	500	625	10	5	151	0.02	0.00	0.61	0.00 ± 0.00	21.08	0.00 ± 0.00	6.42
steinc3_100	500	625	10	5	1519	0.02	0.00	0.61	0.00 ± 0.00	21.14	0.00 ± 0.00	6.33
steinc3_10	500	625	30	5	93	0.01	1.08	0.26	1.08 ± 0.88	21.08	2.15 ± 0.00	6.44
steinc3_100	500	625	30	5	962	0.01	0.62	0.25	0.15 ± 0.26	20.77	0.42 ± 0.36	6.42
steinc4_10	500	625	10	5	115	0.01	0.00	0.24	0.00 ± 0.00	21.20	0.00 ± 0.00	5.05
steinc4_100	500	625	10	5	1148	0.01	0.00	0.23	0.00 ± 0.00	21.29	0.00 ± 0.00	5.12
steinc4_10	500	625	30	5	84	0.01	0.00	0.09	0.00 ± 0.00	21.25	0.00 ± 0.00	5.05
steinc4_100	500	625	30	5	854	0.00	0.00	0.08	0.00 ± 0.00	21.37	0.00 ± 0.00	5.10
steinc5_10	500	625	10	5	258	0.02	0.00	1.03	0.00 ± 0.00	22.21	0.00 ± 0.00	5.30
steinc5_100	500	625	10	5	2600	0.02	0.00	1.03	0.00 ± 0.00	22.03	0.00 ± 0.00	5.29
steinc5_10	500	625	30	5	141	0.01	3.55	0.34	2.20 ± 1.84	21.91	4.73 ± 0.82	5.33
steinc5_100	500	625	30	5	1459	0.01	0.00	0.17	0.21 ± 0.35	22.06	4.18 ± 1.14	5.32
steinc6_10	500	1000	20	5	27	0.01	0.00	0.08	0.00 ± 0.00	0.26	0.00 ± 0.00	11.27
steinc6_100	500	1000	20	5	274	0.01	0.00	0.08	0.00 ± 0.00	0.27	0.00 ± 0.00	11.22
steinc6_10	500	1000	50	5	27	0.01	0.00	0.08	0.00 ± 0.00	0.24	0.00 ± 0.00	11.24
steinc6_100	500	1000	50	5	274	0.01	0.00	0.08	0.00 ± 0.00	0.25	0.00 ± 0.00	11.28
steinc7_10	500	1000	20	5	49	0.01	0.00	0.19	0.00 ± 0.00	4.01	0.00 ± 0.00	11.21
steinc7_100	500	1000	20	5	503	0.01	0.00	0.19	0.00 ± 0.00	3.82	0.00 ± 0.00	11.09
steinc7_10	500	1000	50	5	46	0.01	6.52	0.30	0.00 ± 0.00	7.33	0.00 ± 0.00	11.15
steinc7_100	500	1000	50	5	471	0.01	6.79	0.29	0.00 ± 0.00	7.47	0.00 ± 0.00	11.26
steinc8_10	500	1000	20	5	201	0.02	2.99	2.08	12.49 ± 0.68	45.58	13.10 ± 0.29	10.14
steinc8_100	500	1000	20	5	2102	0.02	7.52	2.04	10.29 ± 1.13	45.69	12.37 ± 0.00	10.09
steinc8_10	500	1000	50	5	104	0.01	0.00	0.19	7.50 ± 2.02	43.20	9.62 ± 0.96	10.19
steinc8_100	500	1000	50	5	1067	0.01	7.87	0.37	8.71 ± 2.54	43.34	11.12 ± 2.30	10.21
steinc9_10	500	1000	20	5	270	0.04	1.48	3.56	2.11 ± 1.06	43.23	4.69 ± 0.21	12.12
steinc9_100	500	1000	20	5	2825	0.04	1.38	3.55	2.59 ± 1.02	43.67	3.46 ± 0.11	12.14
steinc9_10	500	1000	50	5	143	0.02	0.00	0.41	0.07 ± 0.22	45.18	2.56 ± 1.46	12.04
steinc9_100	500	1000	50	5	1492	0.02	1.47	0.81	0.09 ± 0.19	46.07	1.07 ± 0.70	11.93
steinc10_10	500	1000	20	5	331	0.04	2.72	3.31	9.24 ± 1.34	46.78	11.38 ± 1.82	10.60
steinc10_100	500	1000	20	5	3216	0.03	9.76	2.92	14.56 ± 2.08	43.16	19.22 ± 0.85	10.54
steinc10_10	500	1000	50	5	151	0.02	2.65	0.62	9.14 ± 2.37	43.82	17.44 ± 1.53	10.67
steinc10_100	500	1000	50	5	1483	0.02	1.96	0.52	17.48 ± 2.74	44.46	23.92 ± 3.41	10.57
steinc11_10	500	2500	20	5	27	0.01	0.00	0.09	0.00 ± 0.00	0.42	0.00 ± 0.00	18.45
steinc11_100	500	2500	20	5	274	0.01	0.00	0.09	0.00 ± 0.00	0.41	0.00 ± 0.00	18.56
steinc11_10	500	2500	100	5	27	0.01	0.00	0.09	0.00 ± 0.00	0.39	0.00 ± 0.00	18.58
steinc11_100	500	2500	100	5	274	0.01	0.00	0.09	0.00 ± 0.00	0.42	0.00 ± 0.00	18.62
steinc12_10	500	2500	20	5	59	0.02	0.00	0.55	0.00 ± 0.00	0.41	0.00 ± 0.00	18.40
steinc12_100	500	2500	20	5	604	0.02	0.00	0.56	0.00 ± 0.00	0.36	0.00 ± 0.00	18.40
steinc12_10	500	2500	100	5	59	0.02	0.00	0.55	0.00 ± 0.00	0.37	0.00 ± 0.00	18.46
steinc12_100	500	2500	100	5	604	0.02	0.00	0.56	0.00 ± 0.00	0.39	0.00 ± 0.00	18.48
steinc13_10	500	2500	20	5	439	0.15	0.00	16.35	0.00 ± 0.00	2.66	0.00 ± 0.00	18.88
steinc13_100	500	2500	20	5	4463	0.16	0.00	16.54	0.00 ± 0.00	2.54	0.00 ± 0.00	18.83
steinc13_10	500	2500	100	5	236	0.07	2.54	4.44	2.12 ± 1.02	78.24	1.69 ± 0.73	18.68
steinc13_100	500	2500	100	5	2421	0.05	3.55	4.19	2.81 ± 0.79	79.00	3.57 ± 0.25	18.72
steinc14_10	500	2500	20	5	648	0.23	0.00	33.62	0.00 ± 0.00	2.05	0.00 ± 0.00	18.78
steinc14_100	500	2500	20	5	6566	0.23	0.00	34.73	0.00 ± 0.00	2.71	0.00 ± 0.00	18.87
steinc14_10	500	2500	100	5	339	0.08	1.18	8.89	0.00 ± 0.00	75.89	0.10 ± 0.17	18.48
steinc14_100	500	2500	100	5	3364	0.08	3.57	8.43	0.73 ± 1.02	76.78	2.72 ± 1.66	18.63

Table 6: Results for instances SteinC

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc15_10	500	2500	20	5	1162	0.37	0.95	133.63	2.14 ± 1.07	81.59	3.87 ± 0.52	19.22
steinc15_100	500	2500	20	5	11963	0.38	0.96	138.25	1.24 ± 0.51	82.72	2.26 ± 0.69	19.21
steinc15_10	500	2500	100	5	397	0.09	0.76	8.26	5.47 ± 1.60	79.64	6.63 ± 1.48	18.96
steinc15_100	500	2500	100	5	4108	0.10	1.75	8.58	1.18 ± 1.15	80.16	4.84 ± 3.33	18.79
steinc16_10	500	12500	100	5	27	0.17	0.00	1.58	0.00 ± 0.00	2.90	0.00 ± 0.00	258.80
steinc16_100	500	12500	100	5	274	0.18	0.00	1.57	0.00 ± 0.00	2.86	0.00 ± 0.00	259.12
steinc16_10	500	12500	200	5	27	0.18	0.00	1.57	0.00 ± 0.00	2.81	0.00 ± 0.00	258.90
steinc16_100	500	12500	200	5	274	0.17	0.00	1.56	0.00 ± 0.00	2.97	0.00 ± 0.00	258.69
steinc17_10	500	12500	100	5	59	0.36	0.00	5.71	0.00 ± 0.00	4.81	0.00 ± 0.00	259.61
steinc17_100	500	12500	100	5	604	0.36	0.00	6.42	0.00 ± 0.00	3.55	0.00 ± 0.00	259.60
steinc17_10	500	12500	200	5	59	0.36	0.00	5.74	0.00 ± 0.00	2.94	0.00 ± 0.00	259.88
steinc17_100	500	12500	200	5	604	0.36	0.00	6.38	0.00 ± 0.00	3.02	0.00 ± 0.00	259.52
steinc18_10	500	12500	100	5	439	2.99	0.00	265.03	0.00 ± 0.00	35.93	0.00 ± 0.00	266.33
steinc18_100	500	12500	100	5	4463	3.00	0.00	251.58	0.00 ± 0.00	10.22	0.00 ± 0.00	266.16
steinc18_10	500	12500	200	5	439	2.96	0.00	265.07	0.00 ± 0.00	22.55	0.00 ± 0.00	265.96
steinc18_100	500	12500	200	5	4463	2.98	0.00	251.53	0.00 ± 0.00	25.41	0.00 ± 0.00	265.85
steinc19_10	500	12500	100	5	648	4.30	0.00	500.45	0.00 ± 0.00	12.36	0.00 ± 0.00	269.43
steinc19_100	500	12500	100	5	6566	4.39	0.00	528.91	0.00 ± 0.00	14.13	0.00 ± 0.00	269.14
steinc19_10	500	12500	200	5	648	4.26	0.00	499.87	0.00 ± 0.00	35.76	0.00 ± 0.00	269.14
steinc19_100	500	12500	200	5	6566	4.36	0.00	529.06	0.00 ± 0.00	22.82	0.00 ± 0.00	269.22
steinc20_10	500	12500	100	5	1248	8.64	0.00	1846.00	0.00 ± 0.00	50.69	0.00 ± 0.00	278.82
steinc20_100	500	12500	100	5	12533	8.57	0.00	1819.90	0.00 ± 0.00	37.16	0.00 ± 0.00	278.03
steinc20_10	500	12500	200	5	1235	8.17	1.05	4879.20	0.68 ± 0.27	513.04	1.05 ± 0.00	277.82
steinc20_100	500	12500	200	5	12529	8.46	0.03	3509.50	0.02 ± 0.01	108.44	0.03 ± 0.00	278.67
steinc1_10	500	625	10	15	27	0.01	0.00	0.22	0.00 ± 0.00	0.47	0.00 ± 0.00	22.75
steinc1_100	500	625	10	15	274	0.01	0.00	0.22	0.00 ± 0.00	0.45	0.00 ± 0.00	22.74
steinc1_10	500	625	30	15	27	0.01	0.00	0.22	0.00 ± 0.00	0.58	0.00 ± 0.00	22.77
steinc1_100	500	625	30	15	274	0.01	0.00	0.22	0.00 ± 0.00	0.53	0.00 ± 0.00	22.82
steinc2_10	500	625	10	15	59	0.02	0.00	0.51	0.00 ± 0.00	0.51	0.00 ± 0.00	22.82
steinc2_100	500	625	10	15	604	0.02	0.00	0.52	0.00 ± 0.00	0.53	0.00 ± 0.00	22.86
steinc2_10	500	625	30	15	50	0.01	6.00	0.52	2.00 ± 0.00	13.38	2.00 ± 0.00	22.81
steinc2_100	500	625	30	15	533	0.02	2.44	0.57	0.00 ± 0.00	13.29	0.00 ± 0.00	22.78
steinc3_10	500	625	10	15	270	0.07	2.96	7.55	3.15 ± 0.66	106.94	4.20 ± 0.57	23.90
steinc3_100	500	625	10	15	2822	0.07	0.85	7.47	2.35 ± 0.76	110.16	3.00 ± 0.21	24.08
steinc3_10	500	625	30	15	118	0.03	3.39	1.10	4.41 ± 2.46	104.62	6.50 ± 2.13	23.84
steinc3_100	500	625	30	15	1155	0.02	8.14	1.06	10.84 ± 2.59	97.99	11.60 ± 2.13	23.27
steinc4_10	500	625	10	15	308	0.08	2.27	11.10	3.02 ± 1.54	114.26	5.19 ± 0.65	24.06
steinc4_100	500	625	10	15	3219	0.08	0.34	10.90	0.72 ± 1.02	114.20	2.63 ± 1.41	23.77
steinc4_10	500	625	30	15	122	0.03	0.00	0.78	4.10 ± 1.68	105.07	6.56 ± 2.17	23.83
steinc4_100	500	625	30	15	1264	0.03	0.00	0.75	2.47 ± 1.82	104.95	5.91 ± 0.16	23.38
steinc5_10	500	625	10	15	419	0.10	1.67	12.53	2.70 ± 1.51	116.61	6.92 ± 1.49	23.00
steinc5_100	500	625	10	15	4324	0.10	1.19	13.28	0.11 ± 0.25	115.92	0.00 ± 0.00	23.24
steinc5_10	500	625	30	15	143	0.03	11.89	1.45	11.54 ± 2.27	98.21	12.59 ± 1.85	22.76
steinc5_100	500	625	30	15	1481	0.03	10.53	1.41	10.56 ± 2.22	101.36	12.63 ± 1.15	22.69
steinc6_10	500	1000	20	15	27	0.01	0.00	0.21	0.00 ± 0.00	0.58	0.00 ± 0.00	27.14
steinc6_100	500	1000	20	15	274	0.02	0.00	0.21	0.00 ± 0.00	0.77	0.00 ± 0.00	27.20
steinc6_10	500	1000	50	15	27	0.01	0.00	0.21	0.00 ± 0.00	0.76	0.00 ± 0.00	27.19
steinc6_100	500	1000	50	15	274	0.02	0.00	0.21	0.00 ± 0.00	0.60	0.00 ± 0.00	27.11
steinc7_10	500	1000	20	15	59	0.03	0.00	0.61	0.00 ± 0.00	1.17	0.00 ± 0.00	27.23
steinc7_100	500	1000	20	15	604	0.03	0.00	0.58	0.00 ± 0.00	1.22	0.00 ± 0.00	27.08
steinc7_10	500	1000	50	15	59	0.03	0.00	0.56	0.00 ± 0.00	0.70	0.00 ± 0.00	27.07
steinc7_100	500	1000	50	15	604	0.03	0.00	0.55	0.00 ± 0.00	1.02	0.00 ± 0.00	27.13
steinc8_10	500	1000	20	15	293	0.11	3.41	14.45	4.40 ± 1.29	120.35	8.53 ± 0.59	29.87
steinc8_100	500	1000	20	15	2988	0.11	9.40	14.01	5.91 ± 1.51	120.71	8.26 ± 1.30	29.51
steinc8_10	500	1000	50	15	142	0.05	6.34	2.92	6.76 ± 2.49	114.15	13.85 ± 1.08	29.00
steinc8_100	500	1000	50	15	1483	0.05	14.36	2.90	13.40 ± 2.86	107.24	12.99 ± 1.17	29.16

Table 7: Results for instances SteinC (continued)

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc9_10	500	1000	20	15	342	0.12	3.22	17.24	0.61 ± 0.42	119.87	1.46 ± 1.05	27.99
steinc9_100	500	1000	20	15	3588	0.12	2.05	17.03	0.44 ± 0.87	124.41	0.84 ± 0.11	28.85
steinc9_10	500	1000	50	15	160	0.05	7.50	3.62	1.44 ± 1.35	116.41	2.92 ± 1.30	27.78
steinc9_100	500	1000	50	15	1626	0.05	3.01	3.30	4.78 ± 2.25	114.13	5.02 ± 2.65	28.05
steinc10_10	500	1000	20	15	488	0.15	3.48	19.80	1.27 ± 0.97	127.24	3.83 ± 1.36	29.83
steinc10_100	500	1000	20	15	5095	0.15	1.33	20.29	1.24 ± 1.47	129.61	2.53 ± 1.82	29.71
steinc10_10	500	1000	50	15	209	0.06	0.48	3.89	1.67 ± 2.70	115.05	3.99 ± 0.28	29.00
steinc10_100	500	1000	50	15	2356	0.07	2.50	4.42	0.25 ± 0.79	115.33	0.00 ± 0.00	28.96
steinc11_10	500	2500	20	15	27	0.02	0.00	0.31	0.00 ± 0.00	0.70	0.00 ± 0.00	44.16
steinc11_100	500	2500	20	15	274	0.03	0.00	0.31	0.00 ± 0.00	1.20	0.00 ± 0.00	44.06
steinc11_10	500	2500	100	15	27	0.02	0.00	0.31	0.00 ± 0.00	0.88	0.00 ± 0.00	44.22
steinc11_100	500	2500	100	15	274	0.02	0.00	0.31	0.00 ± 0.00	1.12	0.00 ± 0.00	44.01
steinc12_10	500	2500	20	15	59	0.05	0.00	1.24	0.00 ± 0.00	1.43	0.00 ± 0.00	45.23
steinc12_100	500	2500	20	15	604	0.05	0.00	1.30	0.00 ± 0.00	0.64	0.00 ± 0.00	45.45
steinc12_10	500	2500	100	15	59	0.05	0.00	1.24	0.00 ± 0.00	1.37	0.00 ± 0.00	45.25
steinc12_100	500	2500	100	15	604	0.05	0.00	1.30	0.00 ± 0.00	1.47	0.00 ± 0.00	45.11
steinc13_10	500	2500	20	15	439	0.46	0.00	49.36	0.00 ± 0.00	5.91	0.00 ± 0.00	48.85
steinc13_100	500	2500	20	15	4463	0.45	0.00	48.99	0.00 ± 0.00	11.27	0.00 ± 0.00	48.42
steinc13_10	500	2500	100	15	277	0.21	9.03	20.92	5.20 ± 1.46	163.59	7.10 ± 2.21	48.24
steinc13_100	500	2500	100	15	2860	0.19	7.13	19.34	4.61 ± 2.12	167.47	6.63 ± 0.83	47.99
steinc14_10	500	2500	20	15	648	0.65	0.00	92.97	0.00 ± 0.00	6.17	0.00 ± 0.00	46.41
steinc14_100	500	2500	20	15	6566	0.66	0.00	92.21	0.00 ± 0.00	6.54	0.00 ± 0.00	45.42
steinc14_10	500	2500	100	15	369	0.25	1.90	29.95	0.05 ± 0.17	162.84	0.72 ± 1.25	44.68
steinc14_100	500	2500	100	15	3823	0.26	0.60	29.23	0.78 ± 1.19	169.31	2.36 ± 1.19	44.24
steinc15_10	500	2500	20	15	1248	1.30	0.00	299.30	0.00 ± 0.00	20.69	0.00 ± 0.00	47.95
steinc15_100	500	2500	20	15	12533	1.33	0.00	306.81	0.00 ± 0.00	43.12	0.00 ± 0.00	47.51
steinc15_10	500	2500	100	15	493	0.31	4.46	39.38	0.39 ± 0.66	163.01	0.68 ± 0.62	46.59
steinc15_100	500	2500	100	15	5153	0.31	3.92	40.46	0.02 ± 0.05	166.75	1.12 ± 1.32	45.78
steinc16_10	500	12500	100	15	27	0.48	0.00	4.38	0.00 ± 0.00	7.69	0.00 ± 0.00	747.52
steinc16_100	500	12500	100	15	274	0.48	0.00	4.50	0.00 ± 0.00	4.14	0.00 ± 0.00	748.09
steinc16_10	500	12500	200	15	27	0.48	0.00	4.77	0.00 ± 0.00	3.46	0.00 ± 0.00	747.98
steinc16_100	500	12500	200	15	274	0.48	0.00	4.31	0.00 ± 0.00	6.65	0.00 ± 0.00	747.65
steinc17_10	500	12500	100	15	59	0.98	0.00	16.54	0.00 ± 0.00	10.24	0.00 ± 0.00	751.31
steinc17_100	500	12500	100	15	604	0.98	0.00	16.08	0.00 ± 0.00	10.74	0.00 ± 0.00	749.25
steinc17_10	500	12500	200	15	59	0.98	0.00	16.77	0.00 ± 0.00	16.87	0.00 ± 0.00	750.39
steinc17_100	500	12500	200	15	604	0.99	0.00	16.06	0.00 ± 0.00	19.87	0.00 ± 0.00	749.13
steinc18_10	500	12500	100	15	439	8.29	0.00	740.54	0.00 ± 0.00	89.64	0.00 ± 0.00	761.04
steinc18_100	500	12500	100	15	4463	8.03	0.00	709.78	0.00 ± 0.00	41.82	0.00 ± 0.00	759.58
steinc18_10	500	12500	200	15	439	8.26	0.00	731.93	0.00 ± 0.00	35.56	0.00 ± 0.00	759.37
steinc18_100	500	12500	200	15	4463	8.06	0.00	676.60	0.00 ± 0.00	23.45	0.00 ± 0.00	759.73
steinc19_10	500	12500	100	15	648	11.64	0.00	1339.00	0.00 ± 0.00	45.69	0.00 ± 0.00	763.61
steinc19_100	500	12500	100	15	6566	11.84	0.00	1445.10	0.00 ± 0.00	46.52	0.00 ± 0.00	764.48
steinc19_10	500	12500	200	15	648	11.68	0.00	1340.30	0.00 ± 0.00	66.28	0.00 ± 0.00	763.00
steinc19_100	500	12500	200	15	6566	11.81	0.00	1445.40	0.00 ± 0.00	45.00	0.00 ± 0.00	762.63
steinc20_10	500	12500	100	15	1248	23.40	0.00	5108.80	0.00 ± 0.00	58.82	0.00 ± 0.00	781.06
steinc20_100	500	12500	100	15	12533	23.72	0.00	5125.40	0.00 ± 0.00	216.05	0.00 ± 0.00	781.86
steinc20_10	500	12500	200	15	1248	23.37	0.00	4859.90	0.00 ± 0.00	103.91	0.00 ± 0.00	780.99
steinc20_100	500	12500	200	15	12533	23.69	0.00	6113.90	0.00 ± 0.00	175.49	0.00 ± 0.00	780.27
steinc1_10	500	625	10	25	27	0.02	0.00	0.33	0.00 ± 0.00	0.67	0.00 ± 0.00	33.97
steinc1_100	500	625	10	25	274	0.02	0.00	0.33	0.00 ± 0.00	0.57	0.00 ± 0.00	33.67
steinc1_10	500	625	30	25	27	0.02	0.00	0.33	0.00 ± 0.00	0.68	0.00 ± 0.00	33.81
steinc1_100	500	625	30	25	274	0.02	0.00	0.33	0.00 ± 0.00	0.69	0.00 ± 0.00	33.87
steinc2_10	500	625	10	25	59	0.03	0.00	0.80	0.00 ± 0.00	0.65	0.00 ± 0.00	33.83
steinc2_100	500	625	10	25	604	0.03	0.00	0.80	0.00 ± 0.00	1.04	0.00 ± 0.00	33.86
steinc2_10	500	625	30	25	50	0.02	6.00	0.81	2.00 ± 0.00	15.48	2.00 ± 0.00	33.83
steinc2_100	500	625	30	25	533	0.02	2.44	0.86	0.00 ± 0.00	15.41	0.00 ± 0.00	33.86

Table 8: Results for instances SteinC (continued)

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc3_10	500	625	10	25	270	0.11	2.96	11.79	2.96 ± 1.16	140.18	4.07 ± 0.37	35.64
steinc3_100	500	625	10	25	2822	0.10	0.85	11.53	2.79 ± 0.95	143.84	3.22 ± 0.68	35.52
steinc3_10	500	625	30	25	118	0.04	3.39	1.73	5.25 ± 2.11	126.83	4.24 ± 3.06	34.58
steinc3_100	500	625	30	25	1155	0.04	8.14	1.65	8.96 ± 1.78	128.17	10.71 ± 1.56	34.30
steinc4_10	500	625	10	25	314	0.13	2.23	17.78	2.96 ± 1.62	144.82	4.14 ± 1.65	36.46
steinc4_100	500	625	10	25	3219	0.13	1.21	25.85	0.83 ± 0.79	145.25	2.86 ± 0.78	36.62
steinc4_10	500	625	30	25	122	0.05	0.00	1.22	3.44 ± 2.21	131.19	6.83 ± 0.47	34.92
steinc4_100	500	625	30	25	1264	0.05	0.00	1.17	2.82 ± 1.93	125.79	4.46 ± 1.73	36.17
steinc5_10	500	625	10	25	426	0.17	2.58	21.88	3.26 ± 1.84	156.16	5.79 ± 3.13	34.98
steinc5_100	500	625	10	25	4310	0.16	5.15	20.29	2.47 ± 1.77	140.61	4.56 ± 3.11	35.24
steinc5_10	500	625	30	25	143	0.05	11.89	2.25	11.47 ± 1.87	125.48	12.82 ± 1.46	34.49
steinc5_100	500	625	30	25	1481	0.05	10.53	2.18	10.75 ± 2.09	128.79	11.41 ± 1.14	34.06
steinc6_10	500	1000	20	25	27	0.02	0.00	0.31	0.00 ± 0.00	0.65	0.00 ± 0.00	40.56
steinc6_100	500	1000	20	25	274	0.02	0.00	0.31	0.00 ± 0.00	0.80	0.00 ± 0.00	40.53
steinc6_10	500	1000	50	25	27	0.02	0.00	0.31	0.00 ± 0.00	0.78	0.00 ± 0.00	40.55
steinc6_100	500	1000	50	25	274	0.02	0.00	0.31	0.00 ± 0.00	0.68	0.00 ± 0.00	40.50
steinc7_10	500	1000	20	25	59	0.04	0.00	0.93	0.00 ± 0.00	1.72	0.00 ± 0.00	40.40
steinc7_100	500	1000	20	25	604	0.04	0.00	0.90	0.00 ± 0.00	1.56	0.00 ± 0.00	40.28
steinc7_10	500	1000	50	25	59	0.04	0.00	0.88	0.00 ± 0.00	1.65	0.00 ± 0.00	40.40
steinc7_100	500	1000	50	25	604	0.04	0.00	0.85	0.00 ± 0.00	1.73	0.00 ± 0.00	40.17
steinc8_10	500	1000	20	25	294	0.18	5.78	24.52	4.86 ± 2.24	154.59	7.37 ± 1.37	45.34
steinc8_100	500	1000	20	25	3302	0.20	0.24	25.49	0.14 ± 0.28	154.51	0.65 ± 0.70	45.66
steinc8_10	500	1000	50	25	138	0.08	9.42	4.79	11.30 ± 1.71	147.72	17.39 ± 1.45	44.31
steinc8_100	500	1000	50	25	1735	0.09	0.00	2.71	0.21 ± 0.35	150.62	0.00 ± 0.00	43.87
steinc9_10	500	1000	20	25	353	0.21	0.00	14.33	0.11 ± 0.27	163.06	0.00 ± 0.00	42.82
steinc9_100	500	1000	20	25	3556	0.19	1.27	26.15	0.46 ± 0.60	161.38	2.11 ± 1.24	43.72
steinc9_10	500	1000	50	25	160	0.08	5.00	5.64	2.38 ± 2.32	150.57	4.58 ± 1.30	42.67
steinc9_100	500	1000	50	25	1626	0.08	2.95	5.12	4.72 ± 1.69	142.86	6.68 ± 1.01	41.55
steinc10_10	500	1000	20	25	478	0.24	1.26	29.83	3.30 ± 2.25	159.62	5.74 ± 2.19	44.90
steinc10_100	500	1000	20	25	5187	0.26	1.91	34.74	0.47 ± 0.83	161.12	0.05 ± 0.09	44.77
steinc10_10	500	1000	50	25	210	0.10	3.81	5.80	2.00 ± 2.17	146.76	5.40 ± 1.53	43.74
steinc10_100	500	1000	50	25	2422	0.10	2.06	7.05	0.00 ± 0.00	142.88	0.00 ± 0.00	43.91
steinc11_10	500	2500	20	25	27	0.04	0.00	0.50	0.00 ± 0.00	0.87	0.00 ± 0.00	68.56
steinc11_100	500	2500	20	25	274	0.04	0.00	0.49	0.00 ± 0.00	0.58	0.00 ± 0.00	69.07
steinc11_10	500	2500	100	25	27	0.04	0.00	0.50	0.00 ± 0.00	0.55	0.00 ± 0.00	70.56
steinc11_100	500	2500	100	25	274	0.04	0.00	0.49	0.00 ± 0.00	1.31	0.00 ± 0.00	69.05
steinc12_10	500	2500	20	25	59	0.08	0.00	1.97	0.00 ± 0.00	1.98	0.00 ± 0.00	70.49
steinc12_100	500	2500	20	25	604	0.08	0.00	2.03	0.00 ± 0.00	1.89	0.00 ± 0.00	74.61
steinc12_10	500	2500	100	25	59	0.08	0.00	1.96	0.00 ± 0.00	2.32	0.00 ± 0.00	70.33
steinc12_100	500	2500	100	25	604	0.08	0.00	2.05	0.00 ± 0.00	0.72	0.00 ± 0.00	74.20
steinc13_10	500	2500	20	25	439	0.73	0.00	82.66	0.00 ± 0.00	19.00	0.00 ± 0.00	75.59
steinc13_100	500	2500	20	25	4463	0.72	0.00	84.16	0.00 ± 0.00	30.89	0.00 ± 0.00	76.50
steinc13_10	500	2500	100	25	291	0.34	3.78	36.25	1.20 ± 0.96	238.16	1.83 ± 0.79	75.33
steinc13_100	500	2500	100	25	2991	0.33	2.44	36.37	2.02 ± 1.63	237.17	3.35 ± 0.39	74.89
steinc14_10	500	2500	20	25	648	1.01	0.00	147.83	0.00 ± 0.00	11.40	0.00 ± 0.00	71.36
steinc14_100	500	2500	20	25	6566	1.03	0.00	145.08	0.00 ± 0.00	9.44	0.00 ± 0.00	70.61
steinc14_10	500	2500	100	25	369	0.39	1.90	47.32	0.05 ± 0.17	233.86	0.00 ± 0.00	68.60
steinc14_100	500	2500	100	25	3823	0.41	0.60	46.11	0.86 ± 1.19	232.91	1.84 ± 0.97	68.88
steinc15_10	500	2500	20	25	1248	2.02	0.00	468.35	0.00 ± 0.00	40.95	0.00 ± 0.00	73.11
steinc15_100	500	2500	20	25	12533	2.09	0.00	493.36	0.00 ± 0.00	17.30	0.00 ± 0.00	73.20
steinc15_10	500	2500	100	25	510	0.51	1.96	62.85	0.00 ± 0.00	231.98	0.00 ± 0.00	72.53
steinc15_100	500	2500	100	25	5243	0.51	2.86	65.88	0.00 ± 0.00	231.15	0.10 ± 0.17	73.22
steinc16_10	500	12500	100	25	27	0.88	0.00	6.95	0.00 ± 0.00	9.50	0.00 ± 0.00	1232.10
steinc16_100	500	12500	100	25	274	0.81	0.00	6.92	0.00 ± 0.00	8.52	0.00 ± 0.00	1229.60
steinc16_10	500	12500	200	25	27	0.79	0.00	6.99	0.00 ± 0.00	11.63	0.00 ± 0.00	1230.30
steinc16_100	500	12500	200	25	274	0.86	0.00	6.95	0.00 ± 0.00	9.10	0.00 ± 0.00	1232.40

Table 9: Results for instances SteinC (continued)

Instance					Greedy		Destroy-and-Repair		Tabu (2000)		Tabu (10000)	
Name	n	$\|E\|$	b	h	Solution	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds	Improvement (%)	Seconds
steinc17_10	500	12500	100	25	59	1.84	0.00	25.23	0.00 ± 0.00	25.80	0.00 ± 0.00	1234.70
steinc17_100	500	12500	100	25	604	1.67	0.00	24.47	0.00 ± 0.00	20.89	0.00 ± 0.00	1235.00
steinc17_10	500	12500	200	25	59	1.57	0.00	25.27	0.00 ± 0.00	18.94	0.00 ± 0.00	1236.00
steinc17_100	500	12500	200	25	604	1.60	0.00	24.43	0.00 ± 0.00	26.07	0.00 ± 0.00	1533.60
steinc18_10	500	12500	100	25	439	13.11	0.00	1462.30	0.00 ± 0.00	268.74	0.00 ± 0.00	1249.20
steinc18_100	500	12500	100	25	4463	12.89	0.00	1363.90	0.00 ± 0.00	116.55	0.00 ± 0.00	1249.20
steinc18_10	500	12500	200	25	439	13.15	0.00	1112.30	0.00 ± 0.00	49.47	0.00 ± 0.00	1246.60
steinc18_100	500	12500	200	25	4463	12.88	0.00	1083.00	0.00 ± 0.00	95.46	0.00 ± 0.00	1247.10
steinc19_10	500	12500	100	25	648	18.62	0.00	2131.90	0.00 ± 0.00	99.51	0.00 ± 0.00	1252.10
steinc19_100	500	12500	100	25	6566	19.05	0.00	2666.20	0.00 ± 0.00	76.83	0.00 ± 0.00	1250.80
steinc19_10	500	12500	200	25	648	18.62	0.00	2132.70	0.00 ± 0.00	98.14	0.00 ± 0.00	1249.70
steinc19_100	500	12500	200	25	6566	19.08	0.00	2313.40	0.00 ± 0.00	109.82	0.00 ± 0.00	1249.60
steinc20_10	500	12500	100	25	1248	37.41	0.00	9047.00	0.00 ± 0.00	79.46	0.00 ± 0.00	1273.20
steinc20_100	500	12500	100	25	12533	38.19	0.00	7973.80	0.00 ± 0.00	269.95	0.00 ± 0.00	1274.40
steinc20_10	500	12500	200	25	1248	37.42	0.00	9074.30	0.00 ± 0.00	117.52	0.00 ± 0.00	1272.90
steinc20_100	500	12500	200	25	12533	38.14	0.00	8984.00	0.00 ± 0.00	285.56	0.00 ± 0.00	1275.00

Table 10: Results for instances SteinC (continued)

References

Avella, P., Villacci, D. and Sforza, A. (2005). A Steiner arborescence model for the feeder reconfiguration in electric distribution networks, *European Journal of Operational Research* **164**: 505–509.

Balakrishnan, A. and Altinkemer, K. (1992). Using a hop-constrained model to generate alternative communication network design, *ORSA Journal on Computing* **4**: 192–205.

Beasley, J. E. (1990). OR-Library: distributing test problems by electronic mail, *Journal of the Operational Research Society* **41**: 1069–1072.

Canuto, S. A., Resende, M. G. C. and Ribeiro, C. C. (2001). Local search with perturbations for the prize-collecting Steiner tree problem in graphs, *Networks* **38**: 50–58.

Carneiro, M. S., França, P. and Silveira, P. D. (1996). Long-range planning of power distribution systems: secondary networks, *Computers and Electrical Engineering* **22**: 179–191.

Costa, A. M., Cordeau, J.-F. and Laporte, G. (2006a). Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints, Technical Report CRT-2006-14, Center for Research on Transportation, Montreal.

Costa, A. M., Cordeau, J.-F. and Laporte, G. (2006b). Steiner tree problems with profits, *INFOR* **44**: 99–115.

Cunha, A. S., Lucena, A., Maculan, N. and Resende, M. G. C. (2003). A relax and cut algorithm for the prize collecting Steiner problem in graphs, Proceedings of Mathematical Programming in Rio, Búzios - Brazil : 72–78.

Gendreau, M., Hertz, A. and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem, *Management Science* **40**: 1276–1290.

Gouveia, L. (1995). Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints, *Computers & Operations Research* **22**: 959–970.

Gouveia, L. (1996). Multicommodity flow models for spanning trees with hop constraints, *European Journal of Operational Research* **95**: 178–190.

Gouveia, L. (1998). Using variable redefinition for computing lower bounds for minimum spanning and Steiner trees with hop constraints, *INFORMS Journal on Computing* **10**: 180–188.

Gouveia, L. and Magnanti, T. L. (2003). Network flow models for designing diameter-constrained minimum-spanning and Steiner trees, *Networks* **41**: 159–173.

Gouveia, L. and Requejo, C. (2001). A new Lagrangean relaxation approach for the hop-constrained minimum spanning tree problem, *European Journal of Operational Research* **132**: 539–552.

Lawler, E. (1976). *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.

LeBlanc, L., Chifflet, J. and Mahey, P. (1999). Packet routing in telecommunication networks with path and flow restrictions., *INFORMS Journal on Computing* **11**: 188–197.

Ljubić, I., Weiskircher, R., Pferschy, U., Klau, G., Mutzel, P. and Fischetti, M. (2005). Solving the prize-collecting Steiner tree problem to optimality, Proceedings of ALENEX, Seventh Workshop on Algorithm Engineering and Experiments, Vancouver, Canada.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems, *Computers & Operations Research* (forthcoming).

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, *Transportation Science* (forthcoming).

Voß, S. (1999). The Steiner tree problem with hop constraints, *Annals of Operations Research* **86**: 321–345.

Voß, S. (2006). Steiner tree problems in telecomunications, in P. Pardalos and M. G. C. Resende (eds), *Handbooks of Telecommunications*, Springer, New York.